

NO-R182 582

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 8

1/1

USER INTERFACE SUBS (U) GENERAL ELECTRIC CO

SCHENECTADY NY PRODUCTION RESOURCES CONSU

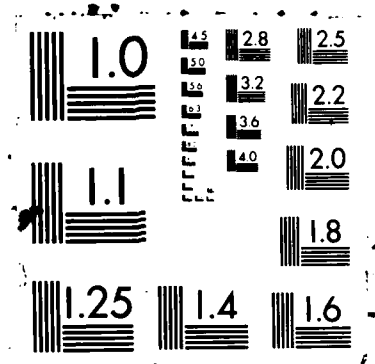
UNCLASSIFIED

L JONES ET AL 01 NOV 85 DS-6201444018

F/G 12/5

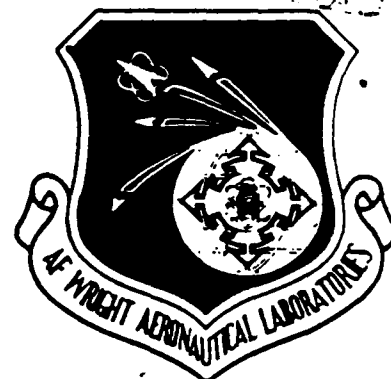
NL

							END						
							8-87						
							DTIC						



AD-A182 582

AFWAL-TR-86-4006
Volume VIII
Part 14



INTEGRATED INFORMATION
SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 14 - Forms Language Compiler Development Specification

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985
November 1985

Approved for public release; distribution is unlimited.

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

DTIC
SELECTED
JUL 16 1987
S & E D

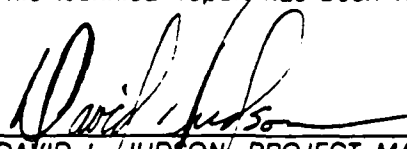
87 7 15 036

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.




DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986

DATE

FOR THE COMMANDER:



GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86

DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

A182 582

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFVAL-TR-86-4006 Vol VIII, Part 14		
6a NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting		6b. OFFICE SYMBOL (If applicable) AFVAL/MLTC	7a. NAME OF MONITORING ORGANIZATION AFVAL/MLTC		
6c ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345			7b. ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6533		
8a NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF		8b. OFFICE SYMBOL (If applicable) AFVAL/MLTC	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5155		
8c ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433			10 SOURCE OF FUNDING NOS		
			PROGRAM ELEMENT NO. 78011F	PROJECT NO. 7500	TASK NO. 62
11. TITLE (Include Security Classification) (See Reverse)					
12. PERSONAL AUTHOR(S) Jones, Larry and Glandorf, Frank					
13a TYPE OF REPORT Final Technical Report		13b TIME COVERED 22 Sept 1980 - 31 July 1985		14. DATE OF REPORT (Yr, Mo, Day) 1985 November	
15. PAGE COUNT 33					
16 SUPPLEMENTARY NOTATION ICAM Project Priority 6201 The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.					
17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB GR			
1308	0905				
19 ABSTRACT (Continue on reverse if necessary and identify by block number) This DS establishes the requirements for the compiler (FLAN) that translates Form Definition Language source files into binary form definition file format.					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21 ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a NAME OF RESPONSIBLE INDIVIDUAL David L. Judson			22b TELEPHONE NUMBER (Include Area Code) 813-255-6976		22c OFFICE SYMBOL AFVAL/MLTC

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

11. Title

Integrated Information Support System (IISS)
Vol VIII - User Interface Subsystem
Part 14 - Forms Language Compiler Development
Specification

A S D 86 0032
9 Jan 1986

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

<u>Subcontractors</u>	<u>Role</u>
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors

Role

Illinois Institute of
Technology

Responsible for factory view
function research (IITRI)
and information models of
small and medium-size business.

North American Rockwell

Reviewer.

Northrop Corporation

Responsible for factory view
function and information
models.

Pritsker and Associates

Responsible for IDEF2 support.

SofTech

Responsible for IDEFO support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors

Role

Boeing Military Aircraft
Company (BMAC)

Responsible for consultation on
applications of the technology
and on IBM computer technology.

Computer Technology
Associates (CTA)

Assisted in the areas of
communications systems, system
design and integration
methodology, and design of the
Network Transaction Manager.

Control Data Corporation
(CDC)

Responsible for the Common Data
Model (CDM) implementation and
part of the CDM design (shared
with DACOM).

D. Appleton Company
(DACOM)

Responsible for the overall CDM
Subsystem design integration
and test plan, as well as part
of the design of the CDM
(shared with CDC). DACOM also
developed the Integration
Methodology and did the schema
mappings for the Application
Subsystems.

Subcontractors

Role

Digital Equipment
Corporation (DEC)

Consulting and support of the
performance testing and on DEC
software and computer systems
operation.

McDonnell Douglas
Automation Company
(McAuto)

Responsible for the support and
enhancements to the Network
Transaction Manager Subsystem
during 1984/1985 period.

On-Line Software
International (OSI)

Responsible for programming the
Communications Subsystem on the
IBM and for consulting on the
IBM.

Rath and Strong Systems
Products (RSSP) (In 1985
became McCormack & Dodge)

Responsible for assistance in
the implementation and use of
the MRP II package (PIOS) that
they supplied.

SofTech, Inc.

Responsible for the design and
implementation of the Network
Transaction Manager (NTM) in
1981/1984 period.

Software Performance
Engineering (SPE)

Responsible for directing the
work on performance evaluation
and analysis.

Structural Dynamics
Research Corporation
(SDRC)

Responsible for the User
Interface and Virtual Terminal
Interface Subsystems.

Other prime contractors under other projects who have
contributed to Test Bed Technology, their contributing
activities and responsible projects are as follows:

<u>Contractors</u>	<u>ICAM Project</u>	<u>Contributing Activities</u>
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

<u>Contractors</u>	<u>ICAM Project</u>	<u>Contributing Activities</u>
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

	<u>Page</u>
SECTION 1.0 SCOPE	1-1
1.1 Identification	1-1
1.2 Functional Summary	1-1
SECTION 2.0 DOCUMENTS	2-1
2.1 Reference Documents	2-1
2.2 Terms and Abbreviations	2-2
SECTION 3.0 REQUIREMENTS	3-1
3.1 Computer Program Definition	3-1
3.1.1 System Capacities	3-1
3.1.2 Interface Requirements	3-1
3.1.2.1 Interface Block Diagram	3-2
3.1.2.2 Detailed Interface Definition	3-4
3.1.2.2.1 Forms Language Compiler Interfaces .	3-4
3.1.2.2.1.1 Form Definition Language	3-4
3.1.2.2.1.2 Binary Form File Format	3-4
3.1.2.2.1.3 User Input to FLAN	3-4
3.1.2.2.1.4 FLAN Error Messages	3-5
3.1.2.2.2 REVFLAN Interfaces	3-6
3.1.2.2.3 MAKINC Interfaces	3-6
3.2 Detailed Functional Requirements	3-6
3.2.1 FLAN Functional Requirements	3-6
3.2.2 REVFLAN Functional Requirements	3-8
3.2.3 MAKINC Functional Requirements	3-8
3.3 Performance Requirements	3-8
3.3.1 Programming Methods	3-8
3.3.2 Program Organization	3-9
3.3.3 Modification Requirements	3-9
SECTION 4.0 QUALITY ASSURANCE PROVISIONS	4-1
4.1 Introduction and Definitions	4-1
4.2 Computer Programming Test and Evaluation .	4-1
SECTION 5.0 PREPARATION FOR DELIVERY	5-1

APPENDICES

A FORM DEFINITION LANGUAGE SYNTAX	A-1
B BINARY FORM DEFINITION FILE RECORD STRUCTURES .	B-1

DS 620144401B
1 November 1985

FIGURES

3-1	FLAN Interfaces	3-3
3-2	FLAN Screen	3-5
3-3	Major Internal Data Structures	3-8

SECTION 1

SCOPE

1.1 Identification

→ This specification establishes the detailed requirements for performance, design, test, and qualification of a computer program identified as the Forms Definition Language Compiler, hereinafter referred to as FLAN. FLAN is one configuration item of the Integrated Information Support System User Interface (IISS UI).

1.2 Functional Summary

→ FLAN is used to compile Form Definition Language source files into binary Form Definition File format. Form Definition files are used as input to the Form Processor in displaying the forms. REVFLAN is used to create a Form Definition Language source file from one or more version 1.0 Form Definition Files which were derived from the DEC FMS utility before FLAN was implemented and which had no source file. MAKINC creates program variable declarations which correspond to the structure of a form and are useful to application programs which make use of Form Processor calls to PDATA and GDATA.

The parser and some procedures of FLAN are used by the Forms Driven Forms Editor (FD FE), the Report Writer (RW), and the Rapid Application Generator (RAP) (all three of which are configuration items). This ensures that the language which is accepted by them is identical.

SECTION 2

DOCUMENTS

2.1 Reference Documents

- [1] ICAM Documentation Standards, ICAM Document IDS 150120000C, 15 September 1983.
- [2] IISS Integration Task Force, Final Report, 1984.
- [3] A. V. Aho and J. D. Ullman, Principles of Compiler Design, Addison-Wesley, 1977.
- [4] S. C. Johnson, "YACC: Yet Another Compiler-Compiler," UNIX* Programmer's Manual, Seventh Edition, Vol. 2, Bell Laboratories, 1983.
- [5] General Electric Co., System Design Specification, 7 February 1983.
- [6] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620144501, 1 November 1985.
- [7] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620144502, 1 November 1985.
- [8] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620144600B, 1 November 1985.
- [9] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620144200B, 1 November 1985.
- [10] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620144700, 1 November 1985.
- [11] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620144401B, 1 November 1985.

* UNIX is a trademark of AT&T Bell Laboratories.

- [12] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620144402B, 1 November 1985.
- [13] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620144100B, 1 November 1985.
- [14] Structural Dynamics Research Corporation, Virtual Terminal Development Specification, DS 620144300B, 1 November 1985.

2.2 Terms and Abbreviations

Application Definition Language: an extension of the Forms Definition Language that includes retrieval of database information and conditional actions. It is used to define interactive application programs.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Display List: is similar to the open list, except that it contains only those forms that have been added to the screen and are currently displayed on the screen.

Field: two-dimensional space on a terminal screen.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition: (FD), forms definition language after compilation. It is read at runtime by the Form Processor.

Forms Definition Language: (FDL), the language in which electronic forms are defined.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor and the Forms Language Compiler.

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Forms Language Compiler: (FLAN), subset of the FE that consists of a batch process that accepts a series of forms definition language statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: instance of forms in windows that are created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Subform: a form that is used within another form.

User Interface: (UI), IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

FLAN is a compiler which translates Form Definition Language source files into binary Form Definition File format. The binary Form Definition Files are then used as input by the Form Processor (another configuration item of the IISS UI) for display and entry of data under the control of other application programs.

While FLAN is normally invoked from the IISS function screen, another version is available which can be invoked from the host system. This second version is required so configuration management software can be used in managing Forms Definition Language files in a manner similar to other source files.

In order to ease the conversion of forms which were not created using the Forms Definition Language, REVFLAN is used. REVFLAN is a program used to create a Forms Definition Language source file from one or more version 1.0 binary Form Definition files which were created using the DEC FMS. The resulting Form Definition Language file may then be FLANed to produce version 2.0 binary Form Definition files. REVFLAN is invoked from the host system.

MAKINC is a program that creates program variable declarations which correspond to the structure of a form and may be used in application programs which make use of the Form Processor calls PDATA and GDATA. The following programming languages are supported: PL/I, COBOL, and C. MAKINC is invoked from the host system.

3.1.1 System Capacities

FLAN is written in the C programming language and runs on a DEC VAX minicomputer under the VMS operating system.

3.1.2 Interface Requirements

FLAN may be invoked from the IISS function screen or from the host system. In either case the user specifies a Forms Definition Language (FDL) source file to be compiled and FLAN will then output binary Form Definition (FD) files. Error messages are directed to the user's terminal.

The format of the binary Form Definition Files produced by FLAN is constrained to agree with the format expected by the Form Processor configuration item.

The syntax of the Form Definition Language accepted as input is based on the preliminary syntax developed by the IISS Integration Task Force and reported in their Final Report. FLAN also accepts statements of the Report Writer language and the Application Generator language but performs no semantic processing on those statements.

REVFLAN is invoked from the host system. The user is prompted for the name of the Forms Definition Language file that REVFLAN will create and for the name of each binary Form Definition file that is to be reverse compiled.

MAKINC is invoked from the host system. The user is prompted for the computer programming language, the name of the output file and for each form for which program variable declarations are desired.

3.1.2.1 Interface Block Diagram

The interface block diagram for FLAN is shown in Figure 3-1. The top box represents the file MYFORMS.FDL which is input to the FLAN compiler (second box). FLAN produces a FD file for each CREATE FORM statement in the source file. Each FD file is input for the Form Processor which is part of the User Interface system. Binary Form Definitions are also input to REVFLAN which produces a FDL file and MAKINC which produces a file with program variable declarations.

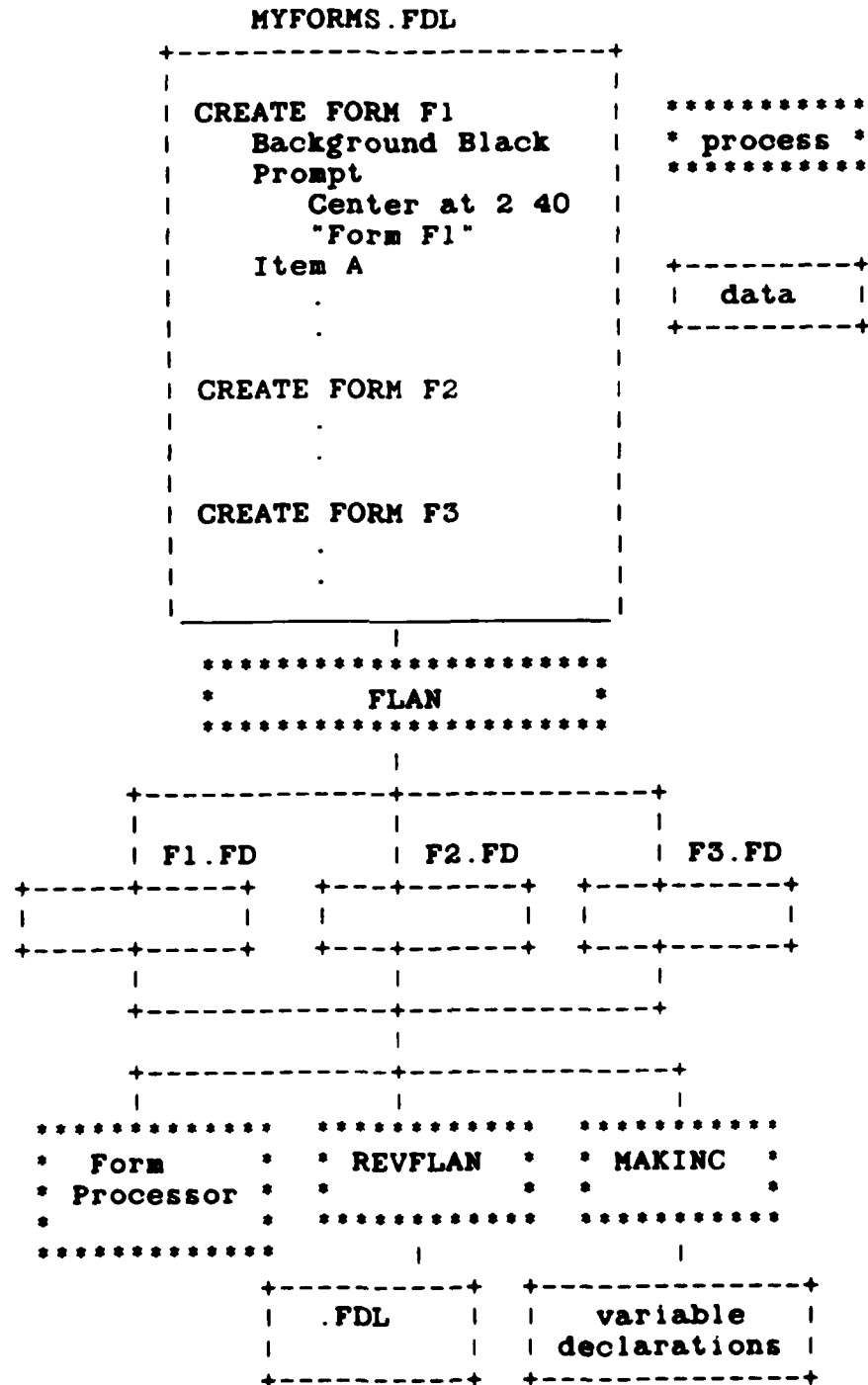


Figure 3-1 FLAN Interfaces

3.1.2.2 Detailed Interface Definition

3.1.2.2.1 Form Language Compiler Interfaces

3.1.2.2.1.1 Form Definition Language

The syntax of the Form Definition Language accepted as input by FLAN is documented in Appendix A. This language is intended to provide access to all Form Processor functionality. It is also intended to be a LALR(1) Grammar for ease of parsing. A number of automatic parser generators are available for LALR(1) grammars, most notably the UNIX* utility YACC.

3.1.2.2.1.2 Binary Form File Format

The formats of the records in the binary Form Definition Files produced as output by FLAN are documented in the include file FFFV2.H, contained in Appendix B. The sequence of records in the file is:

- 1) The version record which identifies the file format.
- 2) The form record for the form.
- 3) A text record for each prompt (both form prompts and field prompts).
- 4) A field record for each field.
- 5) The text of all of the prompts divided into 80 character records.
- 6) The default values of all of the fields divided into 80 character records.

3.1.2.2.1.3 User Input to FLAN

The following is the IISS form FLAN uses to prompt the user for an input file.

*UNIX is a trademark of AT&T Bell Laboratories.

IISS Forms Definition Language Compiler Release 2.0	
Forms Definition Language File Name: _____	
Msg: 0	application

Figure 3-2 FLAN screen

The host system invocation of FLAN is system dependent. The user specifies a Forms Definition Language (FDL) source file to be compiled and FLAN will then output binary Form Definition (FD) files. Error messages are directed to the user's terminal.

3.1.2.2.1.4 FLAN Error Messages

FLAN error messages are of the format:

line number: type - message text

Where line number is the number of the line on which the error occurred. Type is either WARNING, ERROR, or FATAL. WARNING messages do not prevent FD file generation but indicate potential runtime problems. ERROR messages are sufficiently serious to prevent FD file generation but error checking continues for the rest of the file. FATAL messages prevent all further compilation.

3.1.2.2.2 REVFLAN Interfaces

The invocation of REVFLAN is system dependent. The user is prompted for the name of the Forms Definition Language file that REVFLAN will create and for the name of each binary Form Definition file that is to be reverse compiled.

3.1.2.2.3 MAKINC Interfaces

The invocation of MAKINC is system dependent. The user is prompted for the computer programming language, the name of the output file and for each form for which program variable declarations are desired.

3.2 Detailed Functional Requirements

3.2.1 FLAN Functional Requirements

FLAN allows one to specify the following items required by the Form Processor and which are discussed in detail in the Form Processor s Developement Specification:

- 1) Array.
- 2) Attribute.
- 3) Field.
- 4) Form.
- 5) Help form.
- 6) Item.
- 7) Prompt.
- 8) Subform.
- 9) Window.

FLAN is strictly a transformational process -- its sole function is to translate data from one format to another, both of which are well defined. To accomplish this transformation in the most useful manner (where useful is defined as providing functions which will be of use to other CIs which are currently planned), the following internal data structures are used.

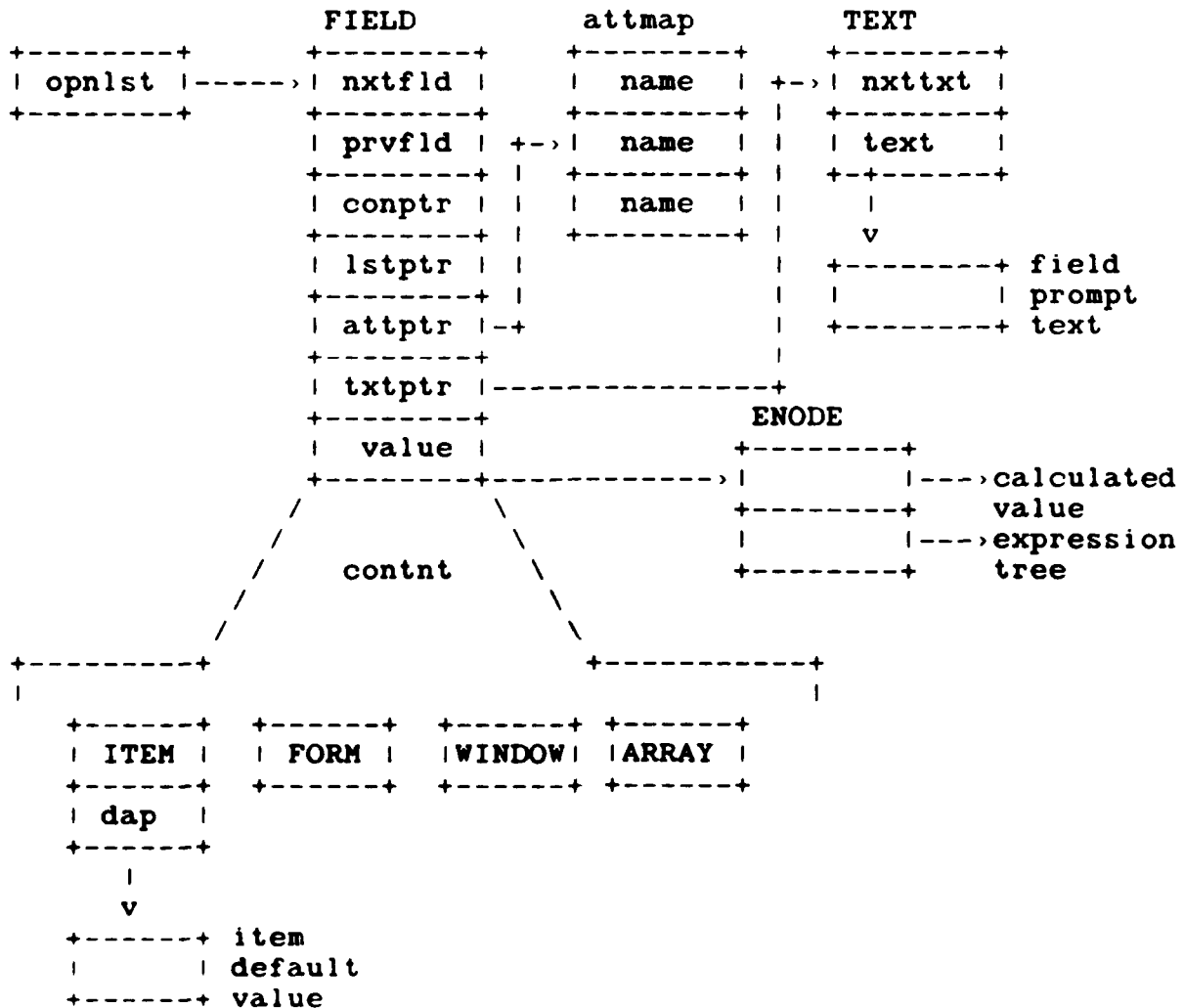


Figure 3-3 Major Internal Data Structures

As statements are read in and recognized, an internal data structure is created and filled in. This data structure is identical to the one used by the Form Processor. Figure 3-2 illustrates this data structure. Note that pointers in the figure which are not shown pointing to anything represent linked lists of items of the type containing the pointer. All of these lists contain zero or more entries depending on the input data.

After all of the input statements have been read in and processed and the internal data structure created and filled in without error, the contents of the internal data structure are recorded in binary form files.

When a processing error does occur, an error message is issued to the terminal specifying as much information as possible about the nature of the error and the location of the input statement causing it. When possible, processing continues after an error is recognized so that additional errors may also be detected.

3.2.2 REVFLAN Functional Requirements

REVFLAN is strictly a transformational process -- its sole function is to translate data from one format to another, both of which are well defined. The input to REVFLAN is constrained to agree with the version 1 binary Form Definition files accepted by the Form Processor. Its output is Form Definition Language source which is syntactically and semantically acceptable input to FLAN. When input to FLAN the output will be equivalent version 2 FD files.

3.2.3 MAKINC Functional Requirements

MAKINC is strictly a transformational process -- its sole function is to translate data from one format to another, both of which are well defined. The input to MAKINC is constrained to agree with version 2 binary Form Definition files accepted by the Form Processor. The output is a set of program variable declarations which are syntactically and semantically correct for the language of choice and which correspond (in structure, name and size) to the forms and fields in the FD file. These programming languages include:

- 1) C.
- 2) COBOL.
- 3) PL/I.

3.3 Performance Requirements

3.3.1 Programming Methods

A parser generator, YACC, is used to create the parser and existing Form Processor routines are used as appropriate as the internal data structures used by fln are identical to that used by the FP.

3.3.2 Program Organization

The actual module structure was refined as other CIs were developed which make use of the functionality of FLAN. FLAN basically consists of a lexical analyzer, an LALR(1) parser, some semantic procedures and a code generator (writes binary Form Definition files). Syntax errors are detected by the lexical analyzer and parser. Semantic errors are detected by the semantic procedures.

3.3.3 Modification Requirements

The use of a parser generator makes changes to the Form Definition Language easy to implement since one specifies only the grammar and not the parse tables. Using existing Form Processor routines ensures that changes made to FP data structures and procedures will require few changes be made to FLAN's procedures.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definitions

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and when they do occur, to make them more noticeable to the programmer and the user. In other words, as much error checking as is practical and possible in each routine should be performed.

4.2 Computer Programming Test and Evaluation

The quality assurance provisions for test consists of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team. Structured design, design walk-through and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

The integration testing entails use of a test application of the VAX. This test program displays forms, reads input from forms, and displays results.

Each function is tested separately, then the entire subsystem is tested as a unit. All testing is done on the IISS testbed VAX.

DS 620144401B
1 November 1985

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the ICAM Integrated Support System (IISS) Test Bed site located at General Electric, Schenectady, New York. The software associated with each CPCI release is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release. Integration with the other IISS CPCI's will be done on the IISS TEST BED on a scheduled basis.

APPENDIX A

FORM DEFINITION LANGUAGE SYNTAX

This document uses the following notation to describe the syntax of the FDL entries:

UPPER-CASE	identifies reserved words that have specific meanings in the FDL. These words are generally required unless the portion of the statement containing them is itself optional.
lower-case	identifies names, numbers, or character strings that the user must supply.
Initial upper-case	identifies a statement or clause that is defined later on.
_ Underscores	Identify reserved words or portions of reserved words that are optional.
{ } Braces	enclosing vertically stacked options indicate that one of the enclosed options is required.
[] Brackets	indicate that the enclosed clause or option is optional. When two or more options are vertically stacked within the brackets, one or none of them may be specified.
... Ellipsis	indicates that the preceding statement or clause may be repeated any number of times.

Form Definition

```
CREATE FORM form_name

    [ SIZE int-1 [ BY int-2 ] ]

    [ BACKGROUND    {BLACK} ]
                        {WHITE}

    [ PROMPT Location prompt_string ... ]

    [ Field_Definition ... ]
```

Field Definition - Items

```
ITEM item_name [ Repeat_Spec ]

    Location

    [ SIZE int-1 [ BY int-2 ] ]

    [ VALUE    { string constant } ]
                  { INDEX(field_name) }
                  { '._TIME' }
                  { '._DATE' }

                  {INPUT }
                  {OUTPUT}
    DISPLAY AS {HIDDEN}
                  {TEXT }

                  [ LEFT ] [ UPPER ]
    [ DOMAIN ( [ RIGHT ] [ LOWER ] [ MUST ENTER ]

                [ MUST FILL ] [ NUMERIC ] [ MAXIMUM int-1]

                [ MINIMUM int-2 ] ) ]

                { help_string }
    [ HELP { help_form_name } ]
                { APPLICATION }

    [ PROMPT Location prompt_string ... ]
```

Field Definition - Forms

FORM form_name [Repeat_Spec]
 Location
 SIZE int-1 [BY int-2]
 [PROMPT Location prompt_string ...]

Field Definition - Windows

WINDOW windows_name [Repeat_Spec]
 Location
 SIZE int-1 [BY int-2]
 {BLACK }
 [BACKGROUND {WHITE }]
 [PROMPT Location prompt_string ...]

Location

```

/
{ [ int ] { LEFT } OF [ field_name ] } +-
{           { RIGHT }           } | AND
{ COLUMN int           } +-

      { [ int ] { BELOW } [ field_name ] } --+
      {           { ABOVE }           } |
      { ROW int           } --+

{ [ int ] { ABOVE } [field_name ] } +-
{           { BELOW }           } | AND
{ ROW int           } +-

[Rpt] AT
      { [ int ] { RIGHT } OF [ field_name ] } --+
      {           { LEFT }           } |
      { COLUMN int           } --+

{ [ int ] { LEFT } OF [ Rpt OF ] [ field_name ] }
{           { RIGHT }           }

{ [ int ] { ABOVE } [ Rpt OF ] [ field_name ] }
{           { BELOW }           }

int-1 int-2 [RELATIVE TO [Rpt OF] [field_name] ]
\

```

Rpt

```

/
TOP LEFT
TOP
TOP RIGHT
LEFT
CENTER
RIGHT
BOTTOM LEFT
BOTTOM
BOTTOM RIGHT
\

```


DS 620144401B
1 November 1985

Repeat Spec

```
+ -
| ( { int-1      } {HORIZONTAL} [ WITH int-3 SPACES ] [ , ... ] ) | - +
| { int-1/int-2 } {VERTICAL  } |
| { int-1/int-1 } |
| { *           } |
| { int-1 / *   } |
+ -
```

APPENDIX B

BINARY FORM DEFINITION FILE RECORD STRUCTURES

```
/* NAME
 *   fffv2.h - Form File Format - Version 2
 *   Written: 18-JUL-1984 13:03:25
 *   Revised: 18-JUL-1984 13:03:25 - SCJONES
 *
 * DESCRIPTION
 *   Record layouts for the binary form definition file
 */

typedef struct      /* version number record */
{   char rectyp;    /* '1' */
    int  vernum;    /* current version number (2) */
    char linefeed;  } VERREC;

typedef struct      /* form record */
{   char  form_name[10]; /* form name */
    char  background[10]; /* background name */
    short row;          /* starting row */
    short col;          /* starting col */
    short width;        /* width */
    short depth;        /* depth */
    short n_txtflds;    /* number of text fields */
    short n_datflds;    /* number of data fields */
    short s_txtbuf;     /* size of the text buffer */
    short s_defbuf;     /* size of the default buffer */
    char  linefeed;    } FRMREC;

typedef struct      /* text record */
{   short row;        /* starting row */
    short col;        /* starting col */
    short len;        /* total length */
    char  linefeed;   } TXTREC;

typedef struct      /* field record */
{   char fld_name[10]; /* field name */
    char fld_type;     /* field type (F, I, W, A) */
    short row;        /* starting row */
    short col;        /* starting col */
    short width;      /* field width */
    short depth;      /* field depth */
    int   min_value;   /* minimum value (if any) */
}
```

DS 620144401B
1 November 1985

```
int    max_value;      /* maximum value (if any) */
char   helpline[80];   /* help text */
char   disp_att[10];   /* display attribute */
short  n_formats;      /* number of formats */
char   format[12][2];  /* format strings */
short  n_arydefs;      /* number of dimensions */
struct /* dimension specification */
{
    char dir;           /* repeat direction (H, V) */
    short cnt;          /* actual repeat count */
    short sp;           /* number of spaces between repetitions */
    short dsp_size;      /* display repeat count */
} array_def[3];        char linefeed; } FLDREC;
```

END

8-87

DTIC